CLAIMS

1. A method for executing a program written for an original computer system on a different host computer system, comprising the steps of:

fetching program code;

translating program code;

emitting translated program code into at least one code cache; and

executing translated code within the at least one code cache in lieu of associated program code when a semantic function of the associated program code is requested.

2. The method of claim 1, wherein the step of fetching program code comprises fetching program instructions with an emulator.

3. The method of claim 2, wherein the emulator is an interpreter/emulator.

4. The method of claim 1, wherein the step of translating the program code comprises translating program instructions with a just-in-time compiler.

5. The method of claim 1, wherein the step of emitting translated program code into at least one code cache comprises emitting translated program code into the at least one code cache via an application programming interface.

6. The method of claim 1, further comprising the step of interpreting and executing program code that has not be emitted into the at least one code cache.

7. The method of claim 6, further comprising the step of emulating actions that would have been performed by the original computer system during execution.

8. The method of claim 1, further comprising the step of, prior to emitting translated program code, growing a code fragment by linking program instructions together.

9. The method of claim 8, wherein the step of linking program instructions together comprises linking program instructions together with a just-in-time compiler.

10. A system for executing program code on a host computer system, comprising:

means for translating program code;

means for emitting translated program code into at least one code cache; and

means for executing translated code within the at least one code cache in lieu of associated program code when a semantic function of the associated program code is requested.

11. The system of claim 10, wherein the means for translating the program code comprise a just-in-time compiler.

12. The system of claim 10, wherein the means for emitting translated program code into at least one code cache comprise an application programming interface.

13. The system of claim 10, further comprising means for interpreting and executing program code.

14. The system of claim 10, further comprising means for emulating actions that would have been performed during execution by an original computer system for which the program code was written.

15. An emulation program configured to emulate an original computer system for which a program was written, the emulation program stored on a computer-readable medium and comprising:

logic configured to translate program code;

logic configured to emit code fragment translations of program code into at least one code cache; and

logic configured to execute the code fragments within the at least one code cache in lieu of associated program code when a semantic function of the associated program code is requested.

16. The program of claim 15, wherein the logic configured to translate the program code comprises a just-in-time compiler.

17. The program of claim 15, wherein the logic configured to emit code fragment translations comprises an application programming interface.

18. The program of claim 15, further comprising logic configured to interpret and execute program code.

19. The program of claim 15, further comprising logic configured to emulate actions of the original computer system for which the program was written.

20. A system for executing program code that was written for an original computer system on a different host computer system, comprising:

an emulator;

a translator;

a virtual machine that comprises a dynamic execution layer interface including a core having at least one code cache in which code fragments can be cached and executed; and

an application programming interface that links the translator to the virtual machine.

21. The system of claim 20, wherein the emulator comprises an interpreter/emulator.

22. The system of claim 20, wherein the translator comprises a just-in-time compiler.

23. The system of claim 20, wherein the application programming interface comprises a set of functions available to the translator including an emit fragment function with which the translator can emit code fragments into the at least one code cache and an execute function with which the translator can request execution of code fragments contained within the at least one code cache.

24. The system of claim 23, wherein the application programming interface is configured such that the emit fragment function can be used to perform at least one of tracking a cached code fragment and associating metadata with a cached code fragment.

25. The system of claim 23, wherein the set of application programming interface functions comprises a lookup fragment function with which cached code fragments can be retrieved.

26. The system of claim 23, wherein the set of application programming interface functions comprises an invalidate fragment function with which individual cached code fragments can be invalidated.

27. The system of claim 23, wherein the set of application programming interface functions comprises a enumerate fragment function with which cached code fragments can be enumerated using metadata associated with the fragments.

28. The system of claim 23, wherein the set of application programming interface functions comprises a cache flush function in which code caches of the dynamic execution layer interface can be flushed.

29. The system of claim 23, wherein the set of application programming interface functions comprises an install callback function with which the translator can be notified as to events that occur within the dynamic execution layer interface.

30. An application programming interface configured to link a translator to a dynamic execution layer interface in an computer system emulating system, comprising:

a set of functions available to the translator including:

an emit fragment function with which the translator can emit code fragments into code caches of the dynamic execution layer interface, and

an execute function with which the translator can request execution of code fragments contained within the at least one code cache.

31. The application programming interface of claim 30, wherein the emit fragment function can be used to perform at least one of tracking a cached code fragment and associating metadata with a cached code fragment.

32. The application programming interface of claim 30, wherein the set of functions further comprises a lookup fragment function with which cached code fragments can be retrieved.

33. The application programming interface of claim 30, wherein the set of functions comprises an invalidate fragment function with which individual cached code fragments can be invalidated.

34. The application programming interface of claim 30, wherein the set of functions comprises a enumerate fragment function with which cached code fragments can be enumerated using metadata associated with the fragments.

35. The application programming interface of claim 30, wherein the set of functions comprises a cache flush function in which code caches of the dynamic execution layer interface can be flushed.

36. The application programming interface of claim 30, wherein the set of functions comprises an install callback function with which the translator can be notified as to particular events that occur within the dynamic execution layer interface.